1. **What database models do you know?**   
   [Hierarchical database model](http://en.wikipedia.org/wiki/Hierarchical_database_model), [network model](http://en.wikipedia.org/wiki/Network_model), [relational model](http://en.wikipedia.org/wiki/Relational_model), [entity–relationship model](http://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model), [object model](http://en.wikipedia.org/wiki/Object_database), [document model](http://en.wikipedia.org/wiki/Document-oriented_database), [entity–attribute–value model](http://en.wikipedia.org/wiki/Entity%E2%80%93attribute%E2%80%93value_model), [star schema](http://en.wikipedia.org/wiki/Star_schema).
2. **Which are the main functions performed by a Relational Database Management System (RDBMS)?**- creating, altering, deleting tables and relationships between them (database schema);  
   - adding, changing, deleting, searching and retrieving of data stored in the tables;  
   - support for the SQL language;  
   - transaction management (optional).
3. **Define what is "table" in database terms.**  
   A [table](http://en.wikipedia.org/wiki/Table_(information)) is a collection of related data held in a structured format within a [database](http://en.wikipedia.org/wiki/Database). It consists of [fields](http://en.wikipedia.org/wiki/Field_(computer_science)) ([columns](http://en.wikipedia.org/wiki/Column_(database))), and [rows](http://en.wikipedia.org/wiki/Row_(database)).   
   In [relational databases](http://en.wikipedia.org/wiki/Relational_database) and [flat file databases](http://en.wikipedia.org/wiki/Flat_file_database), a table is a set of data elements (values) using a model of vertical [columns](http://en.wikipedia.org/wiki/Column_(database)) (which are identified by their name) and horizontal [rows](http://en.wikipedia.org/wiki/Row_(database)), the [cell](http://en.wikipedia.org/w/index.php?title=Cell_(database)&action=edit&redlink=1) being the unit where a row and column intersect. A table has a specified number of columns, but can have any number of rows. Each row is identified by the values appearing in a particular column subset which has been identified as a [unique key](http://en.wikipedia.org/wiki/Unique_key) index.  
   Table is another term for [relation](http://en.wikipedia.org/wiki/Relation_(database)); although there is the difference in that a table is usually a [multiset](http://en.wikipedia.org/wiki/Multiset" \o "Multiset) (bag) of rows where a relation is a set and does not allow duplicates. Besides the actual data rows, tables generally have associated with them some [metadata](http://en.wikipedia.org/wiki/Metadata), such as [constraints](http://en.wikipedia.org/wiki/Check_constraint) on the table or on the values within particular columns.   
   The data in a table does not have to be physically stored in the database. [Views](http://en.wikipedia.org/wiki/View_(database)) are also relational tables, but their data are calculated at query time. Another example are nicknames, which represent a pointer to a table in another database
4. **Explain the difference between a primary and a foreign key.**

|  |  |
| --- | --- |
| Primary key | Foreign key |
| Primary key uniquely identify a record in the table. | Foreign key is a field in the table that is primary key in another table. |
| Primary key can't accept null values. | Foreign key can accept multiple null value. |
| By default, Primary key is clustered index and data in the database table is physically organized in the sequence of clustered index. | Foreign key do not automatically create an index, clustered or non-clustered. You can manually create an index on foreign key. |
| We can have only one Primary key in a table. | We can have more than one foreign key in a table. |

1. **Explain the different kinds of relationships between tables in relational databases.**- one-to-many (or many-to-one): a single record in the first table has many corresponding records in the second table;  
   - many-to-many: records in the first table have many corresponding records in the second one and vice versa;  
   - one-to-one: a single record in a table corresponds to a single record in the other table.
2. **When is a certain database schema normalized? What are the advantages of normalized databases?**Database normalization is the process of organizing the [fields](http://en.wikipedia.org/wiki/Field_(computer_science)) and [tables](http://en.wikipedia.org/wiki/Table_(database)) of a [relational database](http://en.wikipedia.org/wiki/Relational_database) to minimize [redundancy](http://en.wikipedia.org/wiki/Data_redundancy). Normalization usually involves dividing large tables into smaller (and less redundant) tables and defining relationships between them. The objective is to isolate data so that additions, deletions, and modifications of a field can be made in just one table and then propagated through the rest of the database using the defined relationships.
3. **What are database integrity constraints and when are they used?**Data integrity refers to maintaining and assuring the accuracy and consistency of [data](http://en.wikipedia.org/wiki/Data) over its entire [life-cycle](http://en.wikipedia.org/wiki/Information_Lifecycle_Management), and is a critical aspect to the design, implementation and usage of any system which stores, processes, or retrieves data. The term data integrity is broad in scope and may have widely different meanings depending on the specific context – even under the same general umbrella of [computing](http://en.wikipedia.org/wiki/Computing).  
   Data integrity is normally enforced in a [database system](http://en.wikipedia.org/wiki/Database_system) by a series of [integrity constraints](http://en.wikipedia.org/wiki/Integrity_constraints) or rules. Three types of integrity constraints are an inherent part of the relational data model:   
   *-* [*entity integrity*](http://en.wikipedia.org/wiki/Entity_integrity) concerns the concept of a [primary key](http://en.wikipedia.org/wiki/Primary_key). Entity integrity is an integrity rule which states that every table must have a primary key and that the column or columns chosen to be the primary key should be unique and not null;  
   [*- referential integrity*](http://en.wikipedia.org/wiki/Referential_integrity) concerns the concept of a [foreign key](http://en.wikipedia.org/wiki/Foreign_key). The referential integrity rule states that any foreign-key value can only be in one of two states. The usual state of affairs is that the foreign key value refers to a primary key value of some table in the database. Occasionally, and this will depend on the rules of the data owner, a foreign-key value can be [null](http://en.wikipedia.org/wiki/Null_(SQL)). In this case we are explicitly saying that either there is no relationship between the objects represented in the database or that this relationship is unknown;  
   *- domain integrity* specifies that all columns in relational database must be declared upon a defined domain. The primary unit of data in the relational data model is the data item. Such data items are said to be non-decomposable or atomic. A domain is a set of values of the same type. Domains are therefore pools of values from which actual values appearing in the columns of a table are drawn;  
   *- user-defined integrity* refers to a set of rules specified by a user, which do not belong to the entity, domain and referential integrity categories.  
     
   If a database supports these features it is the responsibility of the database to insure data integrity as well as the [consistency model](http://en.wikipedia.org/wiki/Consistency_model) for the data storage and retrieval. If a database does not support these features it is the responsibility of the applications to ensure data integrity while the database supports the [consistency model](http://en.wikipedia.org/wiki/Consistency_model) for the data storage and retrieval.

Having a single, well-controlled, and well-defined data-integrity system increases:  
- stability (one centralized system performs all data integrity operations);  
- performance (all data integrity operations are performed in the same tier as the [consistency model](http://en.wikipedia.org/wiki/Consistency_model));  
- re-usability (all applications benefit from a single centralized data integrity system);  
- maintainability (one centralized system for all data integrity administration).

1. **Point out the pros and cons of using indexes in a database.**Advantages: use an index for quick access to a database table specific information. The index is a structure of the database table the value of one or more columns to sort.  
     
   As a general rule, only when the data in the index column Frequent queries, only need to create an index on the table. The index take up disk space and reduce to add, delete, and update the line speed. In most cases, the speed advantages of indexes for data retrieval greatly exceeds it.   
     
   Disadvantages: too index will affect the speed of update and insert, because it requires the same update each index file. For a frequently updated and inserted into the table, there is no need for a rarely used where the words indexed separately, small table, the cost of sorting will not be great, there is no need to create additional indexes. In some cases, the indexing words may not be fast, for example, the index is placed in a contiguous memory space, which will increase the burden of disk read, which is optimal, it should be through the actual use of the environment to be tested.
2. **What's the main purpose of the SQL language?**SQL (Structured Query Language) is a [special-purpose programming language](http://en.wikipedia.org/wiki/Special-purpose_programming_language) designed for managing data held in a [relational database management system](http://en.wikipedia.org/wiki/Relational_database_management_system) (RDBMS).
3. **What are transactions used for? Give an example.**  
   Transactions are a sequence of database operations which are executed as a single unit:  
   Either all of them execute successfully or none of them is executed at all.  
     
   Example:  
   A bank transfer from one account into another (withdrawal + deposit). If either the withdrawal or the deposit fails the entire operation should be cancelled.
4. **What is a NoSQL database?**A NoSQL or Not Only SQL database provides a mechanism for [storage](http://en.wikipedia.org/wiki/Computer_data_storage) and [retrieval](http://en.wikipedia.org/wiki/Data_retrieval) of data that is modeled in means other than the tabular relations used in [relational databases](http://en.wikipedia.org/wiki/Relational_database).
5. **Explain the classical non-relational data models.**  
   NoSQL databases:  
   - data stored as documents;  
   - single entity (document) is a single record;  
   - documents do not have a fixed structure.
6. **Give few examples of NoSQL databases and their pros and cons.**

NoSQL databases are Redis, MongoDB, CouchDB, Cassandra.  
  
Pros:  
The NoSQL approach presents huge advantages over SQL databases because it allows one to scale an application to new levels. The new data services are based on truly scalable structures and architectures, built for the cloud, built for distribution, and are very attractive to the application developer. There’s no need for DBA, no need for complicated SQL queries and it is fast.  
This is no small matter — a good programmer’s freedom to choose a data model, write a program or an application with familiar tools, reduce dependencies on other people, test and optimize the code without doing guesswork or counting on a black box (DB). Yes, it’s slow on the test system, but someone will take care of it later by tuning the DB…these are all major advantages of the NoSQL movement.  
  
Cons:  
There are some disadvantages to the NoSQL approach. Those are less visible at the developer level, but are highly visible at the system, architecture and operational levels.  
At the system level, data models are key. Not having a skilled authority to design a single, well-defined data model, regardless of the technology used, has its drawbacks. The data model may suffer from duplication of data objects (non-normalized model). This can happen due to the different object model used by different developers and their mapping to the persistency model. At the system level one must also understand the limitations of the chosen data service, whether it is size, ops per second, concurrency model, etc.  
At the architecture level, two major issues are interfaces and interoperability. Interfaces for the NoSQL data services are yet to be standardized. Even DHT, which is one of the simpler interfaces, still has no standard semantics, which includes transactions, none blocking API etc. Each DHT service used comes with its own set of interfaces. Another big issue is how different data structures, such as DHT and a binary tree, just as an example, share data objects. There are no intrinsic semantics for pointers in all those services. In fact, there’s usually not even strong typing in these services — it’s the developer’s responsibility to deal with that. Interoperability is an important point, especially when data needs to be accessed by multiple services. A simple example: backoffice works in Java, web serving works in PHP, can the data be accessed easily from both domains? Clearly one can use web services in front of the data as a data access layer, but that complicates things even more, and reduces business agility, flexibility and performance while increasing development overhead.  
Moving to the operational realm lies the toughest resistance, and rightfully so…The operational environment requires a set of tools that is not only scalable but also manageable and stable, be it on the cloud or on a fixed set of servers. When something goes wrong, it should not require going through the whole chain and up to the developer level to diagnose the problem. In fact, that is exactly what operation managers regard as an operational nightmare.